

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Sparse Caching for Streaming Media

Inventor(s):

Alexandre Grigorovitch

Yejin Choi

Thales de Carvalho

ATTORNEY'S DOCKET NO. MS1-1541US

EV 317722218

BACKGROUND

Media streaming is a process for sending an audio/video presentation and other data from streaming media files or from live streaming sources from one location to another over a network, such as the Internet or an intranet. Typically, media streaming involves sending a streaming media file from a server to a client, where the streaming media file may be presented (viewed and/or listened to) using a media player. Media streaming may also be carried out peer-to-peer. Unlike non-streaming media transfer techniques, which require an entire media file to be transferred before it may be presented, media streaming allows presentation of portions of the streaming media file while it is being transmitted or streamed to the client.

Media streaming may be either unicast, where a streaming media file is streamed from a server to a single client, or multi-cast, where the streaming media file is streamed from a server to multiple clients. Additionally, media streaming may be either live, where a streaming media file including data representing a live event is streamed as it occurs, or on-demand, where the streaming media file is stored in a streaming media file and streamed when it is requested. On-demand media streaming is typically unicast, with a separate streaming event occurring between the server and each client.

A basic streaming media file typically includes at least two streams: a video stream and an audio stream. More complex streaming media file will include multiple video and/or audio streams, each stream being encoded at a different bit rate (i.e., multi-bit rate encoding). For example, a given portion or stream of video may be stored in a multiple bit rate encoded streaming media file in six different video streams, each stream being encoded at a different bit rate. When a client

1 requests the streaming media file from the server, a determination is then made as
2 to the bandwidth of the link between the server and the client. One of the six video
3 streams and an audio stream are then selected for transmission to the client, based
4 on predetermined bandwidth criteria. For example, the video and audio streams
5 may be selected such that their combined bit rates are less than a predetermined
6 percentage of the available link bandwidth. If, at some point in the streaming
7 process, the link bandwidth between the server and the client increases or
8 decreases, a different combination of audio and video streams is then selected to
9 meet the predetermined bandwidth criteria. This type of "stream selection" from a
10 multi-bit rate encoded streaming media file based on available bandwidth is
11 commonly referred to as "intelligent streaming."

12 In some client systems, the streaming experience is enhanced by caching
13 some or all of the received streaming media file at the client prior to playing the
14 streams. Caching the streams prior to playing them reduces the likelihood that a
15 problem in the network connection will interrupt the play of the streaming media
16 file at the client. Furthermore, in some systems, some minimal use of stream
17 navigation (rewinding, replay) may be carried out using the cache.

18 One problem that currently exists with respect to multi-bit rate encoding
19 and/or media stream caching is that there is no way for streams of varying bit rates
20 or non-temporally adjacent streams to be stored and then accessed in a seamless
21 manner, if at all. Typically, when a switch is made between media streams in a
22 streaming media file due to a change in bandwidth, any previously cached portions
23 of the streaming media file are discarded and a new stream cache is established.
24 Similarly, when a jump is made between temporally non-adjacent locations in a
25

1 streaming media file such as a seek operation, any previously cached portions of
2 the streaming media file are discarded and a new stream cache is established.

3 The various systems, methods, and data structures described below address
4 these and other problems.

6 **SUMMARY**

7 Various systems, methods, and data structures are described herein relating
8 to caching streaming media file in a manner that allows for storage and retrieval of
9 portions of the streaming media file that are temporally non-contiguous and/or
10 encoded at differing bit rates.

11 In accordance with one implementation, a caching mechanism is used that
12 employs unique intermediate storage mechanisms that permit such temporally
13 non-contiguous and/or variously encoded portions of a streaming media file to be
14 stored in, and accessed from, a cache file. In accordance with another
15 implementation, a unique cache file structure is used that permits such temporally
16 non-contiguous and/or variously encoded portions of a streaming media file to be
17 stored in, and accessed from, a common file.

19 **BRIEF DESCRIPTION OF THE DRAWINGS**

20 Fig. 1 illustrates an exemplary networking environment in which the
21 various systems, methods, and data structures described herein may be employed.

22 Fig. 2 illustrates exemplary client and server devices.

23 Fig. 3 illustrates an exemplary arrangement and format of media streaming
24 data.

25 Fig. 4 illustrates an exemplary data structure of a cache file.

Fig. 5 illustrates an exemplary operational flow including various acts for recording streaming media.

Fig. 6 illustrates an exemplary operational flow including various acts for playing streaming media.

Fig. 7 illustrates an exemplary general computer environment, which can be used to implement the systems, methods, and data structures described herein.

DETAILED DESCRIPTION

Generally, the various systems, methods, and data structures described herein relate to caching a streaming media file in a manner that allows for storage and retrieval of portions of the streaming media that are temporally non-contiguous and/or encoded at differing bit rates. As will be described, a client side caching mechanism uses unique intermediate storage mechanisms and a unique cache file structure that permits such “dissimilar” portions of a streaming media file to be stored in, and accessed from, a cache file.

Exemplary Environment

Fig. 1 illustrates an exemplary network environment 100. In the environment 100, multiple (x) client devices 102(1), 102(2), . . . , 102(x) are coupled to multiple (y) server devices 104(1), 104(2), . . . , 104(y) via a network 106. The network 106 is intended to represent any of a variety of conventional network topologies and types (including wire and/or wireless networks). The network 106 may include, for example, the Internet, an intranet, a Wide Area Network (WAN), a Local Area Network (LAN), and/or various combinations of any of these or other networks.

1 It is to be appreciated that as used herein, a server device can be any device
2 or process that is a source of a streaming media file, and a client device can be any
3 device or process that receives the streaming media file (e.g., for presentation to a
4 user at the client device). For example, in a peer to peer network, the device or
5 process that is the source of the streaming media file can be referred to as the
6 server device while the device or process that receives the streaming media file
7 can be referred to as the client device.

8 In accordance with the various embodiments described herein,
9 communication between the client devices 102 and the server devices 104 may
10 occur using any of a variety of conventional communication protocols (including
11 public and/or proprietary protocols). In one implementation, communication
12 between devices 102 and 104 occurs using a version of the Hyper Text Transport
13 Protocol (HTTP). In another implementation, communication between devices 102
14 and 104 occurs using the Real Time Streaming Protocol (RTSP). Alternatively,
15 other protocols may be used, such as the Session Initiation Protocol (SIP), the
16 Simple Object Access Protocol (SOAP), and so forth.

17 The devices 102 and 104 may each be any of a variety of conventional
18 computing devices, including desktop PCs, workstations, mainframe computers,
19 Internet appliances, gaming consoles, handheld PCs, cellular telephones, personal
20 digital assistants (PDAs), set-top boxes, etc. One or more of the devices 102 and
21 104 may be the same types of devices, or alternatively different types of devices.

22 Although not shown, one or more additional devices (e.g., firewalls,
23 routers, gateways, bridges, multiple proxy servers, etc.) may be situated between a
24 client device 102 and a server device 104. It should be noted that multiple client
25

1 devices 102 may access a single server device 104 and that a single client device
2 102 may access multiple server devices 146.

3 Fig. 2 illustrates an exemplary server device 104. As shown, the server
4 device 104 includes a streaming module 216 and one or more streaming media
5 files 222. The server device 104 may be, for example, an origin server device 104
6 of Fig. 1, or alternatively another device (e.g., a proxy device). In general, the
7 streaming module 220 functions to receive a request for a streaming media file, or
8 a portion of a streaming media file, such as from the client device 102, and to
9 stream the streaming media file 222 or a portion of the file to the requester.

10 A “media file” includes one or more digital streams of information that may
11 be rendered by a media player. Typically, the media file will include two or more
12 streams that are temporally synchronized. The media file may also include other
13 streams which are independent. The contents of a media file may or may not be
14 compressed. The term “streaming media file” is used herein to indicate that a
15 media file is provided over a network to a client device and that playback of the
16 media file can begin prior to the media file being delivered in its entirety (e.g.,
17 providing the media file data on an as-needed basis rather than pre-delivering the
18 data in its entirety before playback). A streaming media file may be publicly
19 available or alternatively restricted (e.g., restricted to only certain users, available
20 only if the appropriate fee is paid, etc.). A streaming media file can be any of a
21 variety of one or more types of time-based media file, wherein information within
22 the streaming media file specifies the temporal presentation of some or all of the
23 file during playback, such as audio, video, temporal text presentation, animation,
24 etc. Additionally, the streaming media file may be pre-recorded or alternatively
25

1 "live" (e.g., a digital representation of a concert being captured as the concert is
2 performed and made available for streaming shortly after capture).

3 A streaming media file 222 may be stored and/or streamed in accordance
4 with any of a variety of different streaming media file formats. For example, a
5 streaming media file may be stored as a static streaming media file. Alternatively,
6 a streaming media file 222 may be streamed "live" from one or more streaming
7 media sources or producers. In one exemplary implementation, media files are
8 stored and/or streamed in accordance with the ASF format (Advanced Systems
9 Format or Advanced Streaming Format). Additional information regarding ASF is
10 available from Microsoft® Corporation of Redmond, Washington. The same
11 technique can be applied to other formats as well, such as MPEG (Moving
12 Pictures Experts Group)-1, MPEG-2, MPEG-4, Quicktime, etc.

13 As shown, the client device 102 includes a streaming media player 206, a
14 networking module 208, a caching module 210, and one or more cache files 216.
15 Included in the caching module 210 are a media cache module 212 and a byte
16 cache module 214. As shown, the one or more cache files are stored in a data
17 storage module 218 at the client device 102.

18 Generally, the streaming media player 206 provides a means by which a
19 streaming media file 222 may be selected by a user for presented at the client
20 device 102. In accordance with one embodiment, the streaming media player 206
21 is an application or applet that is executed by a processor on the client device 102.
22 The streaming media player 206 is operable to receive and decode streaming
23 media files and to deliver the content of the streaming media files to appropriate
24 video and audio output devices at the client device 102.
25

1 In general, the networking module 208 functions as type a “gateway”
2 between the streaming media player 206 and various sources of streaming media
3 files. In this capacity, the networking module 208 performs a number of functions
4 related to establishing connections between the streaming media player 206 and
5 the various streaming media sources. For example, in accordance with one
6 embodiment, the networking module 208 establishes a network connection
7 between the streaming media player 206 and the server device 104. The
8 networking module 208 also establishes a network connection between the
9 streaming media player 206 and the caching module 210.

10 In addition, the networking module 208 performs a number of functions
11 related to determining from which of a number of available streaming media
12 sources the streaming media player 206 will receive data. For example, in
13 accordance with one embodiment, in response to receiving a request from the
14 streaming media player 206 for all or part of a particular streaming media
15 presentation, the networking module 208 determines whether the request can be
16 satisfied by retrieving the requested data from a previously stored cache file 216,
17 or whether the requested data needs to be retrieved from a server device.

18 In accordance with one embodiment, the networking module 208
19 determines the streaming rate of the streaming media file between the streaming
20 module 220 and the networking module 208 based on a variety of different factors.
21 This can be determined in any of a variety of conventional manners, such as
22 sending test messages between devices 102 and 104, monitoring current and past
23 behavior of connections between devices 102 and 104, receiving an indication of
24 the available bandwidth from streaming module 220, and so forth. Given the
25 current available bandwidth, the networking module 208 initially requests a

1 streaming rate that is a particular amount less than the current available
2 bandwidth. This particular amount can be fixed (e.g., always 50kbps) or dynamic
3 (e.g., 15% of the current available bandwidth, or between 5% and 25% of the
4 current available bandwidth).

5 In accordance with one embodiment, the networking module 208 creates or
6 instantiates the media cache module 212, described below. In accordance with this
7 embodiment, the networking module may create or instantiate a single networking
8 module or multiple networking modules. Other functions that may be performed
9 by the networking module 208 include, without limitation, determining if a cache
10 file related to a particular streaming media presentation is stored in the data
11 storage module and creating cache files in the data storage module. Conventional
12 components that are part of client device 102 may optionally be used to assist the
13 networking module 208. For example, in one exemplary implementation, the
14 Microsoft® Internet Explorer browser program includes cache management
15 functionality such as monitoring the expiration of items in the cache and/or
16 garbage collection, and the networking module 208 may use this functionality in
17 performing the various operations of the networking module outlined above.

18 In accordance with one embodiment, the data storage module 218 is
19 composed of nonvolatile memory. For example, and without limitation, the data
20 storage module 218 may be composed of one or more nonvolatile memory
21 devices, such as magnetic or optical storage devices, magneto optical storage
22 devices, nonvolatile RAM, or other type of nonvolatile storage devices. In
23 accordance with another embodiment, the data storage module 218 is composed of
24 one or more types of volatile memory devices.
25

1 The caching module 210 provides a mechanism by which a streaming
2 media file may be written to and read from a cache file 216 in the data storage
3 module 218. As shown, the caching module 210 includes a media cache module
4 212 and a byte cache module 214. As described in greater detail below, the media
5 cache module 212 and the byte cache module 214 each create and/or manage in
6 one or more intermediate data structures into which various portions and
7 arrangements of received streaming media data are stored.

8 The media cache module 212 provides intermediate data structures for a
9 received streaming media file. In particular, the media cache module 212 creates
10 in one or more computer-readable media five types of data structures, a media
11 cache stream, a media cache segment, a media cache header segment, a byte cache
12 index segment, and a byte cache data segment. In operation, the media cache
13 module 212 creates a single media cache header segment and a number of media
14 cache streams and media cache segments for each media file that is received.

15 The media cache module 212 creates and manages a media cache stream
16 for each different type and encoded bit rate of stream received in a streaming
17 media file by the client device 102. As used herein, a “type” of stream refers to the
18 format or function (e.g. audio or video) of the stream. As will be appreciated, the
19 encoded bit rate of a stream is the bit rate at which the stream was original stored
20 in the streaming media file. As such, if the media cache module 212 receives three
21 video streams from a streaming media file, each having a different bit rate, and
22 two audio streams, each having a different bit rate, the media cache module 212
23 will produce five different media cache streams. In operation, the media cache
24 module 212 will create a new media cache stream each time a new type or bit rate
25 of media stream is received by the media cache module 212.

1 When a stream is received by the media cache module 212, the data within
2 the received stream is stored in a media cache segment within (i.e., logically
3 associated with) a media cache stream of the same type and bit rate as the received
4 stream. For example, if a video stream encoded at a bit rate of X is received by the
5 media cache module 212, the data from the received stream will be stored in a
6 media cache stream of the type “video,” having an associated bit rate of X. If a
7 media cache stream of the type “video” encoded at bit rate X has not yet been
8 created, the media cache module 212 will create such a media cache stream. Once
9 the media cache module 212 has created a media cache stream for a given type
10 and bite rate of a received stream, the media cache module 212 will create a
11 separate media cache segment for each temporally non-contiguous portion of the
12 received stream. As used herein, the term “temporally non-contiguous” refers to
13 portions of a received stream or streaming media file that are not adjacent to one
14 another in terms of the temporal presentation of their content during playback. As
15 such, the media cache module 212 will create a separate media cache segment in a
16 given media cache stream for each portion of the received stream that is not
17 immediately adjacent in time, relative to time structure of the received stream, to
18 another media cache stream in the given media cache stream.

19 Fig. 3 illustrates a graphical representation 300 of an arrangement of media
20 cache streams and media cache segments created by the media cache module 212
21 for a hypothetical streaming media file. The media cache streams and segment are
22 shown as being aligned vertically with respect to a time axis 340, where the time
23 axis indicates the playing time of the streaming media file.

24 As shown, the media cache module 212 has created three separate media
25 cache video streams: media cache video stream (1) 310, media cache video stream

1 (2) 312, and media cache video stream (3) 314, each of which is associated with a
2 different bit rate. Additionally, the media cache module 212 has created two
3 separate media cache audio streams: media cache audio stream (1) 316, and media
4 cache audio stream (2) 318, each of which is associated with a different bit rate.

5 As shown, the media cache module 212 has created: two temporally non-
6 contiguous media cache video segments 320 and 326 within media cache video
7 stream (1) 310; one media cache video segment 322 within media cache video
8 stream (2) 312; and one media cache video segment 324 within media cache video
9 stream (3). Additionally, the media cache module 212 has created: one media
10 cache audio segment 328 within media cache audio stream (1) 316; and one media
11 cache segment 330 within media cache audio stream (2).

12 As noted, the media cache module 212 also creates, for each received
13 streaming media file, a media cache header segment data structure. The media
14 cache header segment includes a file identifier field, a media cache segment count
15 field, one or more media cache segment information fields, and/or a streaming
16 media file description.

17 In accordance with one embodiment, the file identifier field includes a
18 Globally Unique Identifier (GUID) that identifies the cache file into which the
19 data of each of the media cache streams and segments for a given streaming media
20 file are stored. The media cache segment count field stores a number indicating the
21 total number of media cache segments having data stored in the cache file. Each
22 media cache segment information field stores a media cache segment information
23 data structure for each media cache segment having data stored in the cache file.

24 The media cache segment information data structure includes fields
25 containing some or all of the following information about a single media cache

1 segment having data in the cache file: an identifier of the media cache segment; a
2 stream identifier that specifies the media cache stream that includes the media
3 cache segment; a segment start position indicator that indicates the start of the
4 media cache segment in the media cache stream that includes the media cache
5 segment; a segment end position identifier that indicates the end of the media
6 cache segment in the media cache stream that includes the media cache segment; a
7 stream size indicator that specifies the size of the media cache stream including
8 the media cache segment; a previous segment identifier that specifies a media
9 cache segment, if any, immediately preceding the media cache segment in the
10 media cache stream; a next segment identifier that specifies a media cache
11 segment, if any, immediately succeeding the media cache segment in the media
12 cache stream, and a segment data type identifier that specifies the type of data (e.g.
13 audio, video, etc.) included in the media cache segment.

14 The information included in the streaming media file description is
15 dependent on the format of the streaming media file. However, generally, the
16 streaming media file description may include such information as descriptions of
17 the various streams of the media file, a description of the codec used to generate
18 the content of the media file, and/or other meta data related to the content.

19 In operation, the media cache module 212 creates a byte cache index
20 segment and an associated byte cache data segment for each media cache segment
21 created by the media cache module. In accordance with one embodiment, each
22 byte cache data segment includes, without limitation, the presentable data (e.g.,
23 audio or video data, etc.) of the media cache segment from which it is created. In
24 accordance with one embodiment, each byte cache index includes, without
25 limitation, the following information related to the data in its associated byte cache

1 data segment; a start time, a duration time, an offset to the data in the byte cache
2 data segment, and/or the size of the data.

3 In accordance with one embodiment, the media cache module 212 creates
4 or instantiates the byte cache module 214, described below. In accordance with
5 this embodiment, the media cache module 212 may create or instantiate a single
6 byte cache module 214 or multiple byte cache modules.

7 The byte cache module 214 serves to, among other things, serialize and
8 store the data from the byte cache index segments and a byte cache data segments
9 in the cache file 216, in accordance with a predetermined cache file data structure
10 format. Fig. 4 illustrates one such exemplary cache file data structure format for
11 the cache file 216. As shown, the cache file 216 is composed of a number of pages
12 410. Included in these pages are a header page 412 and a number of data pages
13 414 – 420. In general, the data pages 414 – 420 includes the byte cache index and
14 data segments created by the media cache module 212, while the header page 412
15 includes information that describes one or more characteristics of the data pages.

16 As shown in Fig. 4, the header page 412 includes a number of header page
17 data fields 422. Included in the header page data fields 422 are a cache file header
18 data field 424 and a number of cache file control record data fields 426 – 432. The
19 cache file header data field 424 further includes a number of cache header fields
20 436. Included in the cache header fields 436 are a cache header GUID field 438, a
21 flags field 440, a free pages record 442, a number of CFCRs data field 444, and an
22 external block ID field 446. The cache header GUID field 438 includes a globally
23 unique identifier that uniquely identifies the cache file 216. The flags field 440
24 includes an indicator flag that specifies whether the cache file includes valid data.
25

1 The number of CFCRs data field 444 includes an identifier that specifies
2 the number of cache file control records included in the header page data fields
3 422. The external block ID field 446 includes a pointer to a page in the cache file
4 that includes additional cache file control records. The external block ID field 446
5 is used in the instance where the number of cache file records required in the
6 cache file is greater than the number of cache file control records that may be
7 included in the header page 412.

8 The free pages record 442 includes a number of table record data fields
9 441, including a number of free pages data field 443, a first external block ID field
10 445, and a free page record field 447. The number of free pages data field 443
11 includes an identifier indicating the number of free pages in the cache file. The
12 free page record field 447 includes a table particularly identifying the various free
13 data pages in the cache file. In the case where the number of free data pages
14 exceeds the number of free data pages that can be specified in the free page record
15 field 447, the first external block ID includes a pointer to a page that includes a
16 data structure identifying additional free data pages.

17 Each of the cache file control records 426 – 432 is associated with a single
18 byte cache segment. In general, each of the cache file control records 426 – 432
19 includes information defining the position of its associated byte cache in the cache
20 file 216. As shown in Fig. 4, each cache file control record includes a number of
21 CFCR data fields 448. Included in the CFCR data fields 448 are a CFCR GUID
22 field 450, a first page index field 452, a first page offset field 454, a last page
23 offset field 456, and a cache pages table record 458. Each of the CFCR data fields
24 448, and the information contained therein, may be said to be associated with the
25

1 single byte cache segment associated with the cache file control record to which
2 they belong.

3 The GUID field 450 includes a globally unique identifier that uniquely
4 identifies the cache file control record in which it is included. In general, the first
5 page index field 452, the first page offset field 454, and the last page offset field
6 456, include addresses defining a window or range of addresses in which the data
7 from the byte cache segment is stored.

8 In accordance with one embodiment, the first page index field 452 includes
9 an address of the first page in the data pages including its associated byte cache
10 segment. The first page offset field 454 includes a pointer to location in a page,
11 relative to the start of the page indicated by the first page index field 452, of the
12 beginning of the data of its associated byte cache segment. The last page offset
13 field 456 includes a pointer to location in a page, relative to the start of the page
14 indicated by the first page index field 452, of the end of the data of its associated
15 byte cache segment.

16 The cache pages table record 458 includes a number of table record data
17 fields 460, including a number of pages data field 462, a first external block ID
18 464, and a cache page record 466. The number of pages data field 462 includes an
19 identifier indicating the number of data pages that contain data from its associated
20 byte cache segment. The cache page record field 466 includes a table identifying
21 the various data pages that include data from its associated byte cache segment. In
22 the case where the number of data pages including data from its associated byte
23 cache segment exceeds the number of data pages that can be specified in the cache
24 page record field 466, the first external block ID field 464 includes a pointer to a
25

1 page that includes a data structure identifying additional data pages that include
2 data from its associated byte cache segment.

3 Fig. 5 illustrates an exemplary operational flow including various
4 operations 500 for receiving and storing a streaming media file. In accordance
5 with one embodiment, the operational flow is implemented by a client device,
6 such as client device 102 of Fig. 2, and may be performed in software, firmware,
7 hardware, or combinations thereof.

8 As shown in Fig. 5, at the start of the operational flow 500 a request is
9 made 510 for all or part of a specified streaming media file. For example, in
10 accordance with one embodiment, a request is made to a server device, such as
11 server device 104 of Fig. 2, for a particular streaming media file. Following the
12 request 510, a determination operation 512 determines whether the requested
13 streaming media file is available. If it is determined that the requested streaming
14 media file is not available, the operational flow 500 ends. If, however, it is
15 determined that the requested streaming media file is available, a create cache file
16 operation 514 then creates a cache file for storage of the requested streaming
17 media file. In accordance with one embodiment, the create cache file operation
18 514 creates the cache file in a data storage module, such as data storage module
19 218 of Fig. 2.

20 Following the creation of the cache file a request description operation 516
21 requests a description of the streaming media file from the server device. A store
22 description operation 518 then receives and stores the streaming media file
23 description in the created cache file. Next, a select streaming media file stream
24 operation 520 selects a media stream from the server device for transfer to the
25 client device. In accordance with one embodiment, the select streaming media file

1 stream operation 520 selects the stream for transfer based at least in part on the
2 bandwidth currently available between the client device and the server device. A
3 stream determination operation 522 then determines if corresponding media cache
4 stream exists in the cache file for the selected stream. That is, the stream
5 determination operation 522 determines whether a media cache stream having the
6 same type and bit rate as the selected stream is present in the cache file.

7 If it is determined that a corresponding media cache stream does not exists
8 in the cache file for the selected stream, a create media cache stream operation 524
9 then creates a corresponding media cache stream in the cache file. Following the
10 creation of the corresponding media cache stream, a create media cache segment
11 operation 526 creates a media cache segment in the media cache stream created in
12 operation 524 corresponding to the selected stream. Returning to the determination
13 operation 522, if it is determined therein that a corresponding cache stream already
14 exists in the cache file for the selected stream, the operational flow bypasses the
15 create streaming media cache stream operation 524, and proceeds to the create
16 media cache segment operation 526, where a media cache segment is created in
17 the corresponding media cache stream. Following the create media segment
18 operation, a streaming operation 528 then streams data from the selected stream
19 into the created media cache segment.

20 As data from the selected stream is being stream into the created media
21 segment cache, a detect seek operation 530 determines whether a seek operation is
22 being requested. If it is determined that a seek operation is being requested, the
23 operational flow proceeds back to the select streaming media file stream operation
24 520. If it is determined that a seek operation is not being requested, the operational
25 flow proceeds to a detect change in bandwidth operation 532, which determines

1 whether the bandwidth between the client device and the server device has
2 changed since the select streaming media file stream operation 520 was
3 performed. If it is determined that a change in the bandwidth has occurred, the
4 operational flow returns to the stream determination operation 522. If it is
5 determined that a change in the bandwidth has not occurred, the operational flow
6 proceeds to an end of stream determination operation, where it is determined
7 whether the end of the selected stream in the select streaming media file stream
8 operation 520 has been reached. If it is determined that the end of the selected
9 stream has not been reached, the operational flow 500 proceeds back to the detect
10 seek operation 530. If it is determined that the end of the stream selected has been
11 reached the operational flow 500 proceeds to an requested media file complete
12 determination operation 536, where it is determined whether the media file
13 requested at 510 has been completely stored to the cache file. In accordance with
14 one embodiment, the requested media file will be determined to be complete if all
15 desired streams have been completely downloaded at an acceptable bit rate. If it is
16 determined that the media file requested has not been completely stored to the
17 cache file, the operational flow 500 returns to the selected streaming media file
18 stream operation 520. If however is determined that the media file requested has
19 been completely stored to the cache file, the operational flow 500 ends.

20 Fig. 6 illustrates an exemplary operational flow 600 including various
21 operations 600 for retrieving and presenting streaming media. The operational
22 flow 600 includes operation for selectively retrieving a single stream, such as a
23 single video or audio stream, from a cache file, such as cache file 216. It should be
24 appreciated that multiple streams from a streaming media file and/or cache file
25 may be retrieved in accordance with the operational flow 600. In accordance with

1 one embodiment, the operational flow 600 is implemented by a client device, such
2 as client device 102 of Fig. 2, and may be performed in software, firmware,
3 hardware, or combinations thereof.

4 As shown in Fig. 6, at the start of the operational flow 600, a request is
5 received from a data requester (e.g. the media player) to play data from a desired
6 point in a particular type of stream. For example, a request may be received to
7 start playing a video stream from a particular a particular point in time with
8 respect to a particular streaming media file stored in a server. However, rather than
9 going directly to the server to service the request, a preferred bit rate operation 612
10 determines if the requested data is available in the cache file at a preferred bit rate.
11 The preferred bit rate may be set or determined in a number of ways. For example,
12 and without limitation, in accordance with one embodiment, a packet-pair
13 technique is used to determine the preferred bit rate. In accordance with another
14 embodiment, a user may explicitly specify the preferred bit rate.

15 If the requested data is not available in the cache file at the preferred bit
16 rate, a server request operation 614 then requests the requested data from the
17 server and stores the requested data in the cache file. Following the server request
18 operation 614, the operational flow returns to the preferred bit rate operation 612.
19 In accordance with one embodiment, after a number of unsuccessful attempts are
20 made to get the requested data from the server, a determination is made as to
21 whether the requested data is available in the cache file at bit rate lower than the
22 preferred bit rate. If so, the requested data at the lower bit rate may be accessed
23 from the cache file.

24 Next, a start point determination operation 616 determines which media
25 cache stream in the cache file includes the start point of the requested data. A data

1 available operation 618 then determines if the requested data is available from the
2 start point to the end of the stream. Stated another way, the data available
3 operation 618 determines if there are any temporal discontinuities in the data in
4 the stream from the start point to the end of the stream. If it is determined that the
5 data is available from the start point to the end of the stream, the operational flow
6 proceeds to a determine cache segment operation 622, which determines the media
7 cache segment in the determined media cache stream that includes the start point.
8 If, however, it is determined that the data is not available from the start point to the
9 end of the stream, the operational flow proceeds to a server data request operation
10 620, which requests from the server any data that is needed to complete the
11 stream, such that there are no temporal discontinuities from the start point to the
12 end of the stream. The operational flow 600 then proceeds to the determine cache
13 segment operation 622. It should be understood that the operational flow continues
14 on after the request has been made to the server by the server data request
15 operation 620. That is, the operational flow is not suspended until the data
16 requested in operation 620 is received from the server and stored.

17 Following the determine cache segment operation 622, a play operation 624
18 delivers the data from the determined media cache segment to the data requester
19 for play. While the determined media cache segment is being delivered to the data
20 requester, a seek determination operation 626 determines if a seek has been
21 requested to a new start point in the determined media cache stream. This may
22 occur, for example, when a seek operation is performed by the data requester to
23 another temporal location in the determined media cache stream. If the seek
24 determination operation 626 determines that a seek has been requested, the
25 operational flow 600 returns to the data available operation 618. If, however, the

1 seek determination operation 626 determines that a seek has not been requested,
2 and after the data from the determined media cache segment has been completely
3 delivered to the data requester, a cache stream end determination operation 628
4 determines whether the end of the determined media cache stream has been
5 reached. If it is determined that the end of the determined media cache stream has
6 not been reached, a next cache segment operation 630 determines the next media
7 cache segment in the determined media cache stream that is to be accessed. If,
8 however, it is determined that the end of the determined media cache stream has
9 been reached, the operational flow 600 proceeds to an end of file (EOF)
10 determination operation 632, where it is determined if the end of the cache file has
11 been reached. If it is determined that the end of the cache file has not been
12 reached, the operational flow 600 returns to the cache stream end determination
13 operation 628. If, however, it is determined that the end of the cache file has been
14 reached, the operational flow 600 ends.

15 Various operational flows have been illustrated in figs. 5 and 6. It should
16 be noted that the operations illustrated in figures can be performed in the order
17 shown, or alternatively in different orders.

18 Fig. 7 illustrates a general computer environment 700, which can be used to
19 implement the techniques described herein. The computer environment 700 is only
20 one example of a computing environment and is not intended to suggest any
21 limitation as to the scope of use or functionality of the computer and network
22 architectures. Neither should the computer environment 700 be interpreted as
23 having any dependency or requirement relating to any one or combination of
24 components illustrated in the exemplary computer environment 700.
25

1 The computer environment 700 includes a general-purpose computing
2 device in the form of a computer 702. The computer 702 may be, for example, a
3 client device 102 or server device 104 of Figs. 1 or 2. The components of the
4 computer 702 may include, but are not limited to, one or more processors or
5 processing units 704, a system memory 706, and a system bus 708 that couples
6 various system components including the processor 704 to the system memory
7 706.

8 The system bus 708 represents one or more of any of several types of bus
9 structures, including a memory bus or memory controller, a peripheral bus, an
10 accelerated graphics port, and a processor or local bus using any of a variety of
11 bus architectures. By way of example, such architectures can include an Industry
12 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an
13 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)
14 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a
15 Mezzanine bus.

16 The computer 702 typically includes a variety of computer-readable media.
17 Such media can be any available media that is accessible by the computer 702 and
18 includes both volatile and non-volatile media, removable and non-removable
19 media.

20 The system memory 706 includes computer-readable media in the form of
21 volatile memory, such as random access memory (RAM) 710, and/or non-volatile
22 memory, such as read only memory (ROM) 712. A basic input/output system
23 (BIOS) 714, containing the basic routines that help to transfer information
24 between elements within the computer 702, such as during start-up, is stored in
25

1 ROM 712. RAM 710 typically contains data and/or program modules that are
2 immediately accessible to and/or presently operated on by the processing unit 704.

3 The computer 702 may also include other removable/non-removable,
4 volatile/non-volatile computer storage media. By way of example, Fig. 7
5 illustrates a hard disk drive 716 for reading from and writing to a non-removable,
6 non-volatile magnetic media (not shown), a magnetic disk drive 718 for reading
7 from and writing to a removable, non-volatile magnetic disk 720 (e.g., a “floppy
8 disk”), and an optical disk drive 722 for reading from and/or writing to a
9 removable, non-volatile optical disk 724 such as a CD-ROM, DVD-ROM, or other
10 optical media. The hard disk drive 716, magnetic disk drive 718, and optical disk
11 drive 722 are each connected to the system bus 708 by one or more data media
12 interfaces 726. Alternatively, the hard disk drive 716, magnetic disk drive 718, and
13 optical disk drive 722 can be connected to the system bus 708 by one or more
14 interfaces (not shown).

15 The disk drives and their associated computer-readable media provide non-
16 volatile storage of computer-readable instructions, data structures, program
17 modules, and other data for computer 702. Although the example illustrates a hard
18 disk 716, a removable magnetic disk 720, and a removable optical disk 724, it is to
19 be appreciated that other types of computer-readable media which can store data
20 that is accessible by a computer, such as magnetic cassettes or other magnetic
21 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or
22 other optical storage, random access memories (RAM), read only memories
23 (ROM), electrically erasable programmable read-only memory (EEPROM), and
24 the like, can also be utilized to implement the exemplary computing system and
25 environment.

1 A number of program modules may be stored on the hard disk 716,
2 magnetic disk 720, optical disk 724, ROM 712, and/or RAM 710, including by
3 way of example, an operating system 726, one or more application programs 728,
4 other program modules 730, and program data 732.

5 A user can enter commands and information into the computer 702 via
6 input devices such as a keyboard 734 and a pointing device 736 (e.g., a “mouse”).
7 Other input devices 738 (not shown specifically) may include a microphone,
8 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and
9 other input devices are connected to the processing unit 704 via input/output
10 interfaces 740 that are coupled to the system bus 708, but may be connected by
11 other interface and bus structures, such as a parallel port, game port, or a universal
12 serial bus (USB).

13 A monitor 742 or other type of display device can also be connected to the
14 system bus 708 via an interface, such as a video adapter 744. In addition to the
15 monitor 742, other output peripheral devices can include components such as
16 speakers (not shown) and a printer 746 which can be connected to computer 702
17 via the input/output interfaces 740.

18 The computer 702 may operate in a networked environment using logical
19 connections to one or more remote computers, such as a remote computing device
20 748. By way of example, the remote computing device 748 may be a personal
21 computer, portable computer, a server, a router, a network computer, a peer device
22 or other common network node, and the like. The remote computing device 748 is
23 illustrated as a portable computer, and may include many or all of the elements
24 and features described herein relative to computer 702.
25

1 Logical connections between the computer 702 and the remote computer
2 748 are depicted as a local area network (LAN) 750 and a general wide area
3 network (WAN) 752. Such networking environments are commonplace in offices,
4 enterprise-wide computer networks, intranets, and the Internet.

5 When implemented in a LAN networking environment, the computer 702
6 may be connected to a local network 750 via a network interface or adapter 754.
7 When implemented in a WAN networking environment, the computer 702 may
8 include a modem 756 or other means for establishing communications over the
9 wide network 752. The modem 756, which may be internal or external to
10 computer 702, may be connected to the system bus 708 via the input/output
11 interfaces 740 or other appropriate mechanisms. It is to be appreciated that the
12 illustrated network connections are exemplary and that other means of establishing
13 communication link(s) between the computers 702 and 748 may be employed.

14 In a networked environment, such as that illustrated with the computing
15 environment 700, program modules depicted relative to the computer 702, or
16 portions thereof, may be stored in a remote memory storage device. By way of
17 example, remote application programs 758 reside on a memory device of remote
18 computer 748. For purposes of illustration, application programs and other
19 executable program components such as the operating system are illustrated herein
20 as discrete blocks, although it is recognized that such programs and components
21 may reside at various times in different storage components of the computing
22 device 702, and are executed by the data processor(s) of the computer.

23 Various modules and techniques may be described herein in the general
24 context of computer-executable instructions, such as program modules, executed
25 by one or more computers or other devices. Generally, program modules include

1 routines, programs, objects, components, data structures, etc. that perform
2 particular tasks or implement particular abstract data types. Typically, the
3 functionality of the program modules may be combined or distributed as desired in
4 various embodiments.

5 An implementation of these modules and techniques may be stored on or
6 transmitted across some form of computer-readable media. Computer-readable
7 media can be any available media that can be accessed by a computer. By way of
8 example, and not limitation, computer-readable media may comprise “computer
9 storage media” and “communications media.”

10 “Computer storage media” includes volatile and non-volatile, removable
11 and non-removable media implemented in any method or technology for storage
12 of information such as computer-readable instructions, data structures, program
13 modules, or other data. Computer storage media includes, but is not limited to,
14 RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM,
15 digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic
16 tape, magnetic disk storage or other magnetic storage devices, or any other
17 medium which can be used to store the desired information and which can be
18 accessed by a computer.

19 “Communication media” typically embodies computer-readable
20 instructions, data structures, program modules, or other data in a modulated data
21 signal, such as carrier wave or other transport mechanism. Communication media
22 also includes any information delivery media. The term “modulated data signal”
23 means a signal that has one or more of its characteristics set or changed in such a
24 manner as to encode information in the signal. By way of example, and not
25 limitation, communication media includes wired media such as a wired network or

1 direct-wired connection, and wireless media such as acoustic, RF, infrared, and
2 other wireless media. Combinations of any of the above are also included within
3 the scope of computer-readable media.

4 Although the description above uses language that is specific to structural
5 features and/or methodological acts, it is to be understood that the systems and
6 methods defined in the appended claims are not limited to the specific features or
7 acts described. Rather, the specific features and acts are disclosed as exemplary
8 forms of implementing the invention.